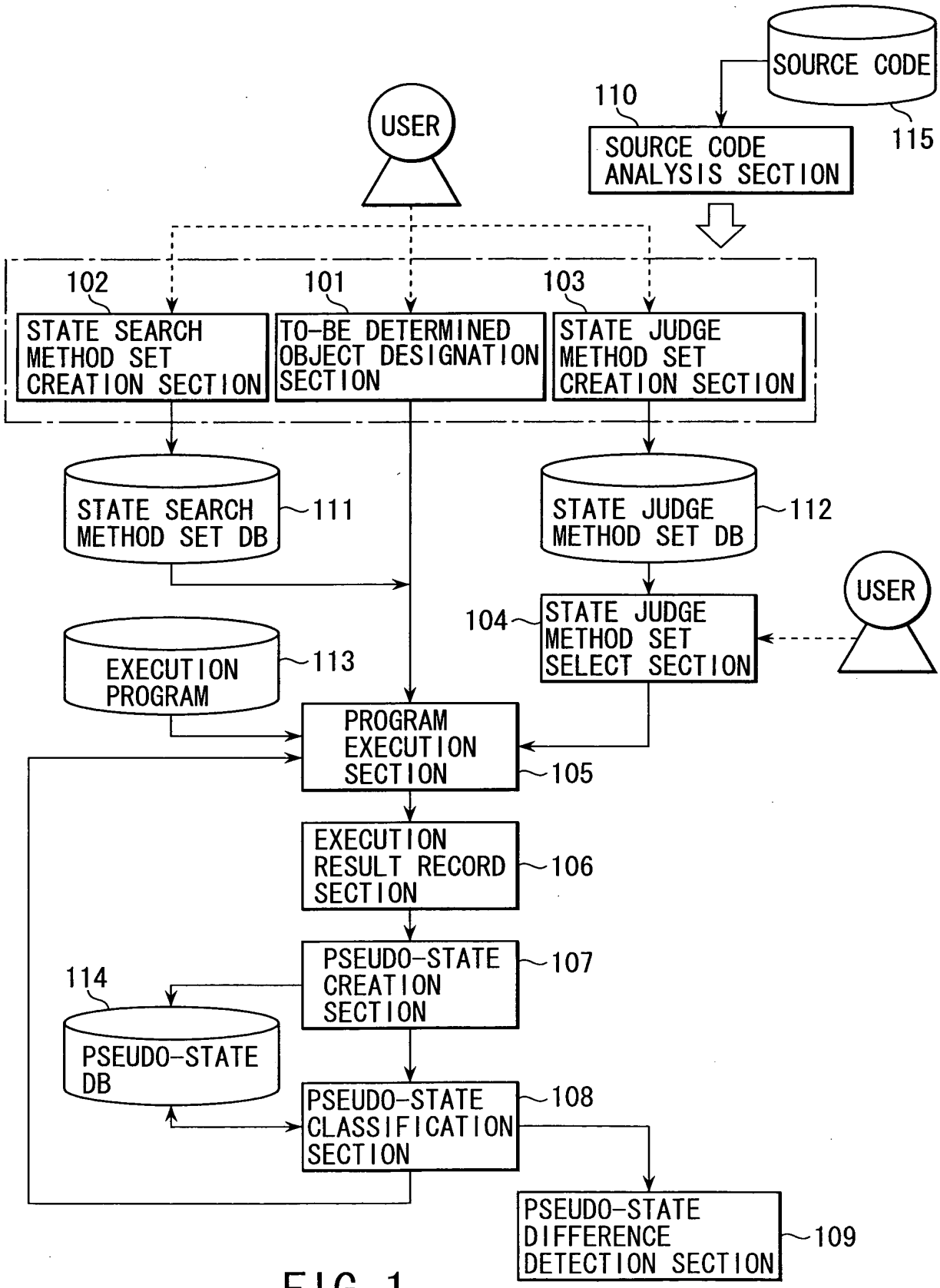

SOURCE CODE
115
110
SOURCE CODE ANALYSIS SECTION
USER
102
101
103
STATE SEARCH METHOD SET CREATION SECTION
TO-BE DETERMINED OBJECT DESIGNATION SECTION
STATE JUDGE METHOD SET CREATION SECTION
STATE SEARCH METHOD SET DB
111
STATE JUDGE METHOD SET DB
112
USER
104
STATE JUDGE METHOD SET SELECT SECTION
EXECUTION PROGRAM
113
PROGRAM EXECUTION SECTION
105
EXECUTION RESULT RECORD SECTION
106
PSEUDO-STATE CREATION SECTION
107
114
PSEUDO-STATE DB
PSEUDO-STATE CLASSIFICATION SECTION
108
PSEUDO-STATE DIFFERENCE DETECTION SECTION
109

FIG. 1

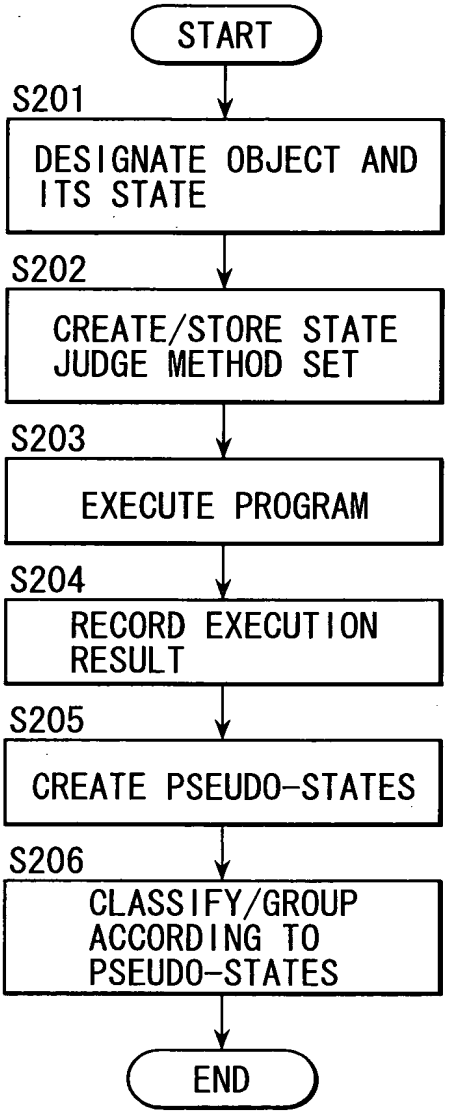

START
S201
DESIGNATE OBJECT AND ITS STATE
S202
CREATE/STORE STATE JUDGE METHOD SET
S203
EXECUTE PROGRAM
S204
RECORD EXECUTION RESULT
S205
CREATE PSEUDO-STATES
S206
CLASSIFY/GROUP ACCORDING TO PSEUDO-STATES
END

FIG. 2

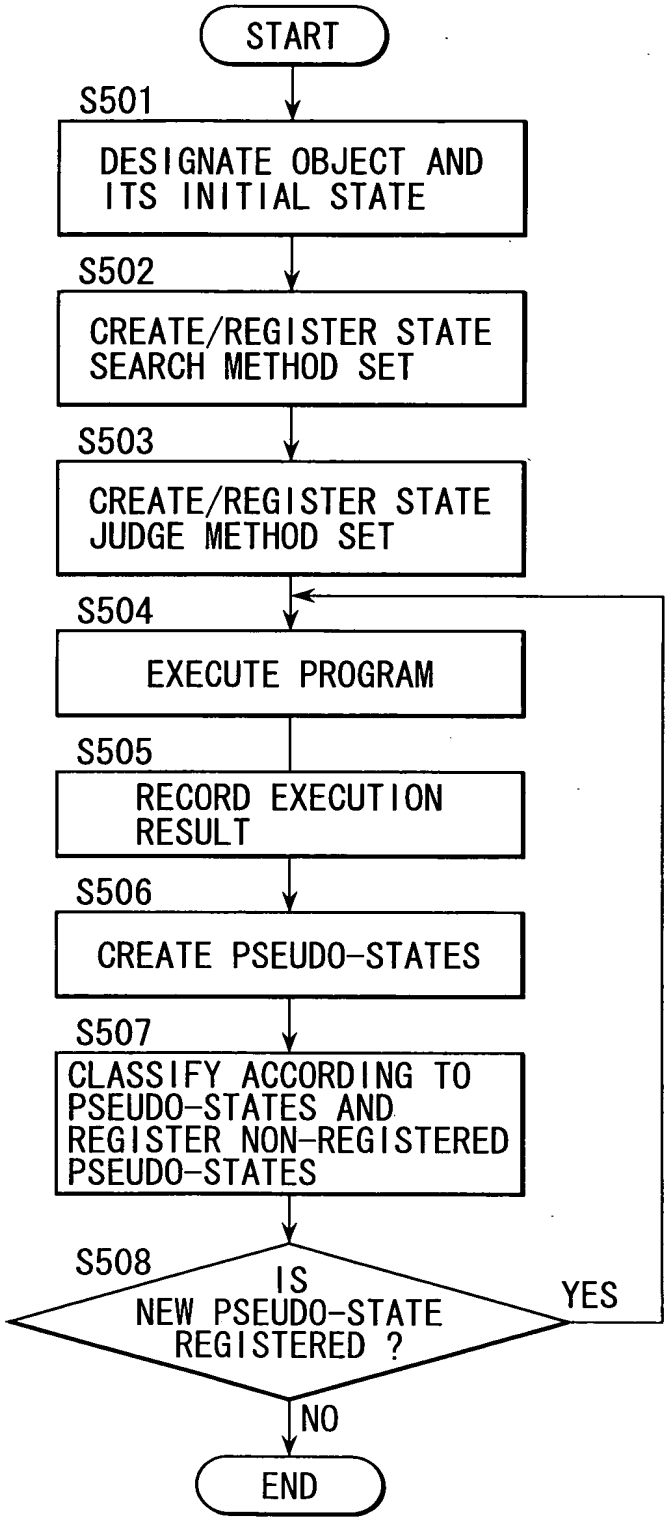

START
S501
DESIGNATE OBJECT AND ITS INITIAL STATE
S502
CREATE/REGISTER STATE SEARCH METHOD SET
S503
CREATE/REGISTER STATE JUDGE METHOD SET
S504
EXECUTE PROGRAM
S505
RECORD EXECUTION RESULT
S506
CREATE PSEUDO-STATES
S507
CLASSIFY ACCORDING TO PSEUDO-STATES AND REGISTER NON-REGISTERED PSEUDO-STATES
S508
IS NEW PSEUDO-STATE REGISTERED ?
YES
NO
END

FIG. 5

FIG. 3

```
//
class Door
{

protected:

    int    angle;
    bool   flgLock;

public:
    void   setAngle(int a);
    void   lock();
    void   unLock();
    bool   isOpened();
};
void Door::setAngle(int a)
{
    if(flgfLock) {
       return;                    //BLOCK 11
    }
    if (a < 0) {
       angle = 0;                 //BLOCK 12
    } else if (a <= 90) {
       angle = a;                 //BLOCK 13
    } else {
       angle = 90;                //BLOCK 14
    }
}
void Door::lock()
{
    if (angle) {
        angle = 0;                //BLOCK 21
    }
    flgLock = true;               //BLOCK 22
}

void Door::unLock()
{
    flgLock = false;              //BLOCK 31
}

bool Door::isOpened()
{
    if (angle == 0) {
        return false;             //BLOCK 41
    } else  {
        return true;              //BLOCK 42
    }
}
//
```

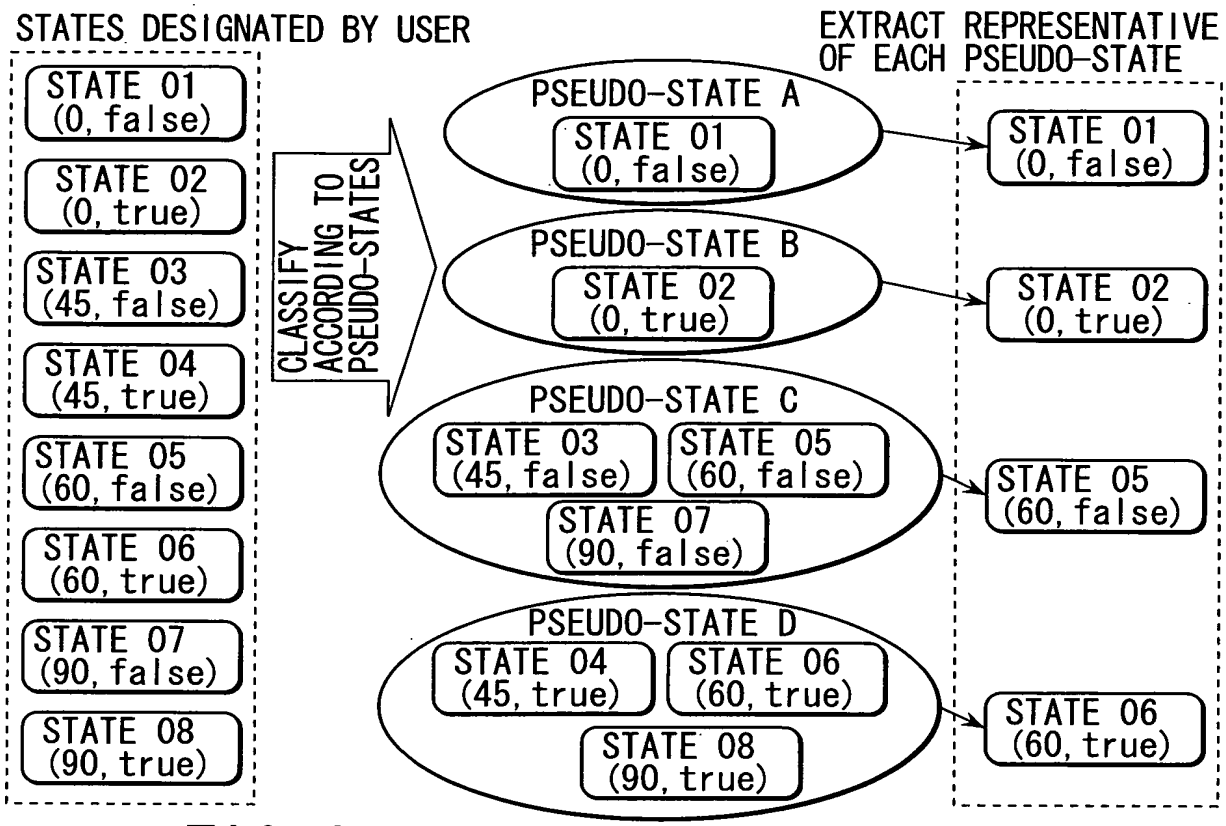

STATES DESIGNATED BY USER
STATE 01 (0, false)
STATE 02 (0, true)
STATE 03 (45, false)
STATE 04 (45, true)
STATE 05 (60, false)
STATE 06 (60, true)
STATE 07 (90, false)
STATE 08 (90, true)
CLASSIFY ACCORDING TO PSEUDO-STATES
PSEUDO-STATE A
STATE 01 (0, false)
PSEUDO-STATE B
STATE 02 (0, true)
PSEUDO-STATE C
STATE 03 (45, false)
STATE 05 (60, false)
STATE 07 (90, false)
PSEUDO-STATE D
STATE 04 (45, true)
STATE 06 (60, true)
STATE 08 (90, true)
EXTRACT REPRESENTATIVE OF EACH PSEUDO-STATE
STATE 01 (0, false)
STATE 02 (0, true)
STATE 05 (60, false)
STATE 06 (60, true)

FIG. 4

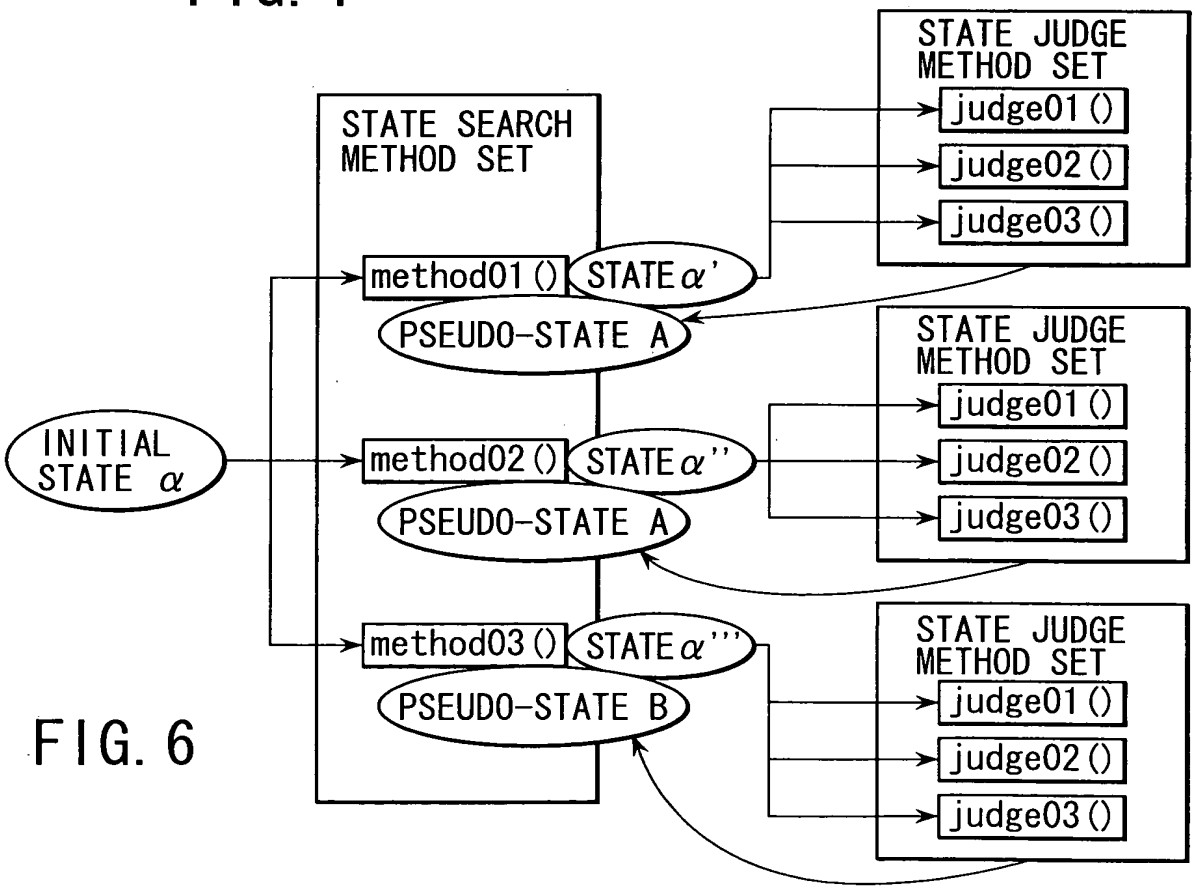

INITIAL STATE α
STATE SEARCH METHOD SET
method01 ()
STATE α'
PSEUDO-STATE A
method02 ()
STATE α''
PSEUDO-STATE A
method03 ()
STATE α'''
PSEUDO-STATE B
STATE JUDGE METHOD SET
judge01 ()
judge02 ()
judge03 ()
STATE JUDGE METHOD SET
judge01 ()
judge02 ()
judge03 ()
STATE JUDGE METHOD SET
judge01 ()
judge02 ()
judge03 ()

FIG. 6

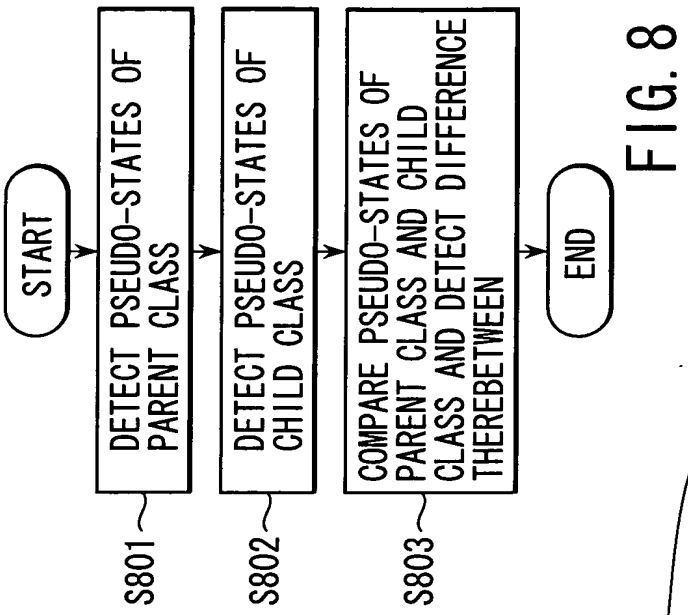

START
DETECT PSEUDO-STATES OF PARENT CLASS
DETECT PSEUDO-STATES OF CHILD CLASS
COMPARE PSEUDO-STATES OF PARENT CLASS AND CHILD CLASS AND DETECT DIFFERENCE THEREBETWEEN
END
S801
S802
S803

FIG. 8

CONSTRUCTOR
PSEUDO-STATE A
method01
PSEUDO-STATE A
method02
PSEUDO-STATE A
method03
PSEUDO-STATE B
FIRST
method01
PSEUDO-STATE C
method02
PSEUDO-STATE B
method03
PSEUDO-STATE D
SECOND
method01
PSEUDO-STATE C
method02
PSEUDO-STATE B
method03
PSEUDO-STATE A
THIRD
method01
PSEUDO-STATE E
method02
PSEUDO-STATE A
method03
PSEUDO-STATE A
FOURTH
method01
PSEUDO-STATE A
method02
PSEUDO-STATE A
method03
PSEUDO-STATE B
FIFTH

FIG. 7

```
class Parent
{
    int p1;
    int p2;
    void PMethod01();
    void PMethod02();
    void PMethod03();
};
class Child:public Parent
{
    void CMethod01();
    void CMethod02();
};
```

INSTANCE	INITIAL STATE	STATE SEARCH METHOD SET	STATE JUDGE METHOD SET	DETECTED PSEUDO-STATES
Parent	p1=0, p2=0	{ PMethod01(), PMethod02(), PMethod03() }	{ PMethod01(), PMethod02(), PMethod03() }	PSEUDO-STATE A, PSEUDO-STATE B
Child	p1=0, p2=0	{ CMethod01(), CMethod02(), }		PSEUDO-STATE A, PSEUDO-STATE B, PSEUDO-STATE C

FIG. 9

FIG. 10

```
EXAMPLE OF CONCRETE STATE
  angle = 30,
  flgLock = true
}
```

FIG. 11

```
EXAMPLE OF STATE JUDGE METHOD SET
  {METHOD:setAngle, ARGUMENT:a=10} ,
  {METHOD:setAngle, ARGUMENT:a=45} ,
  {METHOD:lock, ARGUMENT(NONE), METHOD:isOpened, ARGUMENT(NONE)} ,
  {METHOD:isOpened, ARGUMENT(NONE)}
}
```

FIG. 12

```
EXAMPLE OF PSEUDO-STATE
  {METHOD:setAngle, ARGUMENT:a=10} , EXECUTION RESULT:Path01} ,
  {METHOD:setAngle, ARGUMENT:a=45} , EXECUTION RESULT:Path02} ,
  {METHOD:lock, ARGUMENT(NONE), METHOD:isOpened, ARGUMENT(NONE)} , EXECUTION RESULT:Path03} ,
  {METHOD:isOpened, ARGUMENT(NONE)} , EXECUTION RESULT:Path04}
}
```

FIG. 13

STATE 01	Angle = 0	flgLock = false
STATE 02	Angle = 0	flgLock = true
STATE 03	Angle = 45	flgLock = false
STATE 04	Angle = 45	flgLock = true
STATE 05	Angle = 60	flgLock = false
STATE 06	Angle = 60	flgLock = true
STATE 07	Angle = 90	flgLock = false
STATE 08	Angle = 90	flgLock = true

FIG. 14

```
STATE JUDGE METHOD SET  JudgeSet1 {
    {METHOD:  Door::isOpened,   ARGUMENT:(NONE)} ,
    {METHOD:  Door::unLock,     ARGUMENT:(NONE)} ,
    {METHOD:  Door::lock,       ARGUMENT:(NONE)} ,
    {METHOD:  Door::setAngle,   ARGUMENT:a=-10} ,
    {METHOD:  Door::setAngle,   ARGUMENT:a=45} ,
    {METHOD:  Door::setAngle,   ARGUMENT:a=100}
```

FIG. 17

NAME OF PSEUDO-STATE	STATE
PSEUDO-STATE A	STATE 01
PSEUDO-STATE B	STATE 02
PSEUDO-STATE C	STATE 03, 05, 07
PSEUDO-STATE D	STATE 04, 06, 08

FIG. 18

1. INITIAL STATE α → method01 () → judge01 ()
2. INITIAL STATE α → method01 () → judge02 ()
3. INITIAL STATE α → method01 () → judge03 ()
4. INITIAL STATE α → method02 () → judge01 ()
5. INITIAL STATE α → method02 () → judge02 ()
6. INITIAL STATE α → method02 () → judge03 ()
7. INITIAL STATE α → method03 () → judge01 ()
8. INITIAL STATE α → method03 () → judge02 ()
9. INITIAL STATE α → method03 () → judge03 ()

FIG. 15

```
PSEUDO-STATE ASSOCIATED WITH STATE 01
    { { METHOD:  Door::isOpened,  ARGUMENT:(NONE)} , EXECUTION PATH:BLOCK 41} ,
    { { METHOD:  Door::unLock,    ARGUMENT:(NONE)} , EXECUTION PATH:BLOCK 31} ,
    { { METHOD:  Door::Lock,      ARGUMENT:(NONE)} , EXECUTION PATH:BLOCK 22} ,
    { { METHOD:  Door::setAngle,  ARGUMENT: a=-10} , EXECUTION PATH:BLOCK 12} ,
    { { METHOD:  Door::setAngle,  ARGUMENT: a= 45} , EXECUTION PATH:BLOCK 13} ,
    { { METHOD:  Door::setAngle,  ARGUMENT: a=100} , EXECUTION PATH:BLOCK 14}
  }
```

NAME OF STATE	isOpened ARGUMENT :(NONE)	unLock ARGUMENT :(NONE)	lock ARGUMENT :(NONE)	setAngle ARGUMENT :-10	setAngle ARGUMENT :45	setAngle ARGUMENT :100	NAME OF PSEUDO-STATE
STATE 01	41	31	22	12	13	14	PSEUDO-STATE A
STATE 02	41	31	22	11	11	11	PSEUDO-STATE B
STATE 03	42	31	21-22	12	13	14	PSEUDO-STATE C
STATE 04	42	31	21-22	11	11	11	PSEUDO-STATE D
STATE 05	42	31	21-22	12	13	14	PSEUDO-STATE C
STATE 06	42	31	21-22	11	11	11	PSEUDO-STATE D
STATE 07	42	31	21-22	12	13	14	PSEUDO-STATE C
STATE 08	42	31	21-22	11	11	11	PSEUDO-STATE D

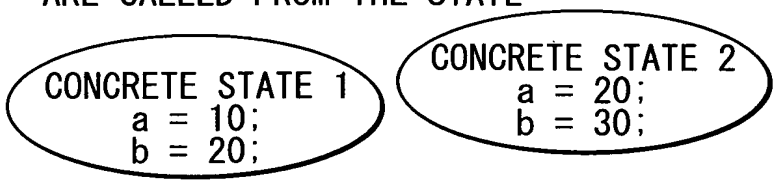
FIG. 16

FIG. 19A
(PRIOR ART)

```
class DUMMY
{
    int a;
    int b;
    void methodA();
    void methodB();
};
```

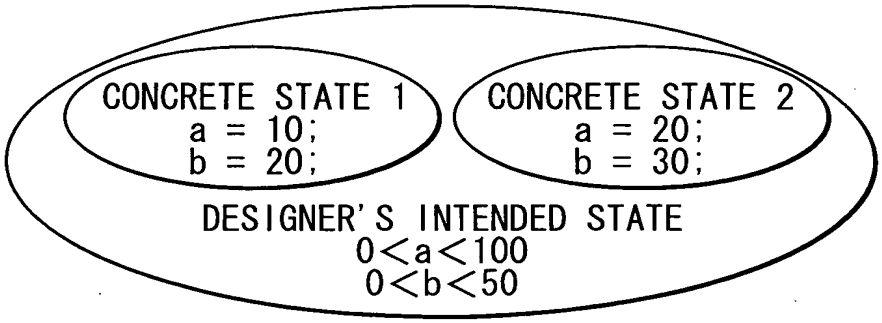
FIG. 19B
(PRIOR ART)

TESTS IN CASE WHERE DESIGNER'S INTENDED STATE IS NOT DEFINED
CONCRETE STATE IS SET AND VARIOUS METHODS ARE CALLED FROM THE STATE



TEST CASE 1:CALL METHOD A FROM CONCRETE STATE 1
TEST CASE 2:CALL METHOD B FROM CONCRETE STATE 1
TEST CASE 3:CALL METHOD A FROM CONCRETE STATE 2
TEST CASE 4:CALL METHOD B FROM CONCRETE STATE 2

TESTS IN CASE WHERE DESIGNER'S INTENDED STATE IS DEFINED



TO ENHANCE EFFICIENCY OF TESTS, IT IS SUFFICIENT, AS FIRST STEP, TO CONDUCT TESTS BASED ON CONCRETE STATE 1, WHICH IS CONCRETE EXAMPLE OF "DESIGNER'S INTENDED STATE"
TEST CASE 1:CALL METHOD A FROM CONCRETE STATE 1
TEST CASE 2:CALL METHOD B FROM CONCRETE STATE 1

FIG. 19C (PRIOR ART)